PATENT APPLICATION

TECHNIQUES FOR GROUPING CIRCUIT ELEMENTS INTO LOGIC BLOCKS

Inventor(s):   Ketan Padalia, a citizen of Canada, residing at
5 Kimbergate Way
Thornhill, Ontario, L4J-6R6 Canada

Kimberly Bozman, a citizen of Canada, residing at
246 Milan Street
Toronto, Ontario, M5A-3Z6 Canada

Vaughn Betz, a citizen of Canada, residing at
233 Riverdale Drive
Toronto, Ontario, M6S-4A8 Canada

Assignee: Altera Corporation
101 Innovation Drive
San Jose, CA, 95134

Entity: Large

TECHNIQUES FOR GROUPING CIRCUIT ELEMENTS INTO LOGIC BLOCKS

BACKGROUND OF THE INVENTION

5   **[0001]**   The present invention relates to techniques for grouping circuits into logic blocks, and more particularly, to techniques for grouping lookup tables, registers and other circuits into logic blocks that allow the circuits to be separated and moved independently during placement of a circuit design for a programmable integrated circuit.

[0002]   A field programmable gate array (FPGA) is a programmable integrated circuit.
10   Programmable integrated circuits also include programmable logic devices (PLDs), programmable logic arrays (PLAs), configurable logic arrays, etc. Many programmable integrated circuits are hybrids of FPGAs and ASICs (application specific integrated circuits).

[0003]   Programmable integrated circuits typically include configurable logic blocks, programmable routing resources, and programmable input/output (I/O) blocks. Configurable
15   logic blocks typically contain combinatorial components such as multiplexers and lookup tables as well as sequential components such as registers. Logic blocks implement user defined logic functions.

[0004]   Lookup tables are the basic logic circuit in many of today's FPGAs. A lookup table (LUT) includes memory cells that can store the truth tables of an arbitrary function. A LUT
20   with k inputs can implement a function with k input variables. The binary value of the input signals are used as a memory address to access one of the LUT memory locations. The LUT outputs the bits that are stored in the accessed memory location.

[0005]   Lookup tables and other circuits in the FPGA are selected to implement equivalent functions performed by circuits in the user design. This selection process involves synthesis
25   and technology mapping. During technology mapping, networks of logic gates in the user-made design are transformed into circuits such as LUTs and registers that can be placed onto the FPGA.

[0006]   The user-made design is converted into a netlist. The netlist pattern organizes the LUTs and registers into logic blocks. The LUTs and registers cannot be subsequently
30   removed from the logic blocks without violating design rules of the FPGA. The logic blocks

are typically grouped into clusters of logic blocks based on routing and timing issues. The clusters of logic blocks are often called logic array blocks (LABs).

[0007]   After synthesis and technology mapping, a placement tool places each of the LABs generated during synthesis to particular LABs on the FPGA. A routing tool then configures the programmable routing resources on the FPGA to connect the LABs together according to the requirements of the user design.

[0008]   During placement, routability and timing issues between logic blocks become apparent that were not apparent during synthesis. For example, a synthesis tool may group a LUT and a register in the same logic block. During placement, it may become clear that routability would be improved if the LUT and the register were placed into different logic blocks on an FPGA.

[0009]   However, a placement tool cannot separate the LUT and register without potentially violating design rules of the FPGA. FPGAs have design rules that limit which LUTs and registers can be placed into the same logic block and the same LAB. The design rules are constraints imposed by the particular architecture of the FPGA.

[0010]   Therefore, it would be desirable to provide techniques for grouping LUTs and registers into logic blocks so that they can be more flexibly rearranged without violating design rules of the programmable integrated circuit.

BRIEF SUMMARY OF·THE INVENTION
[0011]   The present invention provides techniques for grouping circuits elements (e.g., LUTs and registers) into logic blocks on a programmable integrated circuit (IC). A packing tool of the present invention organizes each of the circuit elements into separate abstract blocks. Each LUT, register, and other circuit element is grouped into a separate abstract block. The abstract blocks are then grouped into logic blocks based on routing and timing concerns.

[0012]   Subsequently, a determination is made whether placement information indicates that a design goal would be improved by rearranging at least a portion of the user design. The user design can be rearranged by placing one of the abstract blocks into a different logic block.

[0013]   Individual abstract blocks can be separated and moved independently of other circuits that were previously grouped into the same logic block. For example, a LUT and register that were previously placed into the same logic block can be separated. The LUT and the register can be placed into different logic blocks to further improve routability of the design and signal timing.

[0014]   Other objects, features, and advantages of the present invention will become apparent upon consideration of the following detailed description and the accompanying drawings, in which like reference designations represent like features throughout the figures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015]   Figure 1 illustrates a generalized process for mapping a user made circuit design to a programmable integrated circuit that can be modified to implement techniques of the present invention;

[0016]   Figure 2 graphically illustrates how a packing tool can map an abstract block to a logic block and a cluster of logic blocks according to an embodiment of the present invention;

[0017]   Figure 3 illustrates a process for grouping logic blocks into logic blocks and clusters of logic blocks according to the principles of the present invention;

[0018]   Figure 4 illustrates an example of a logic block with a lookup table that can be used to according to the techniques of the present invention;

[0019]   Figure 5 illustrates an example of a process for grouping abstract blocks into logic blocks and separating abstract blocks from the logic blocks to improve design criteria according to an embodiment of the present invention;

[0020]   Figure 6 is a simplified block diagram of a programmable logic device that can implement embodiments of the present invention; and

[0021]   Figure 7 is a block diagram of an electronic system that can implement embodiments of the present invention.

3

DETAILED DESCRIPTION OF THE INVENTION

[0022] Figure 1 is flow chart that illustrates a generalized process for mapping a user made circuit design to a programmable integrated circuit (IC) such as an FPGA or a PLD. The first step 102 in this mapping process involves synthesis and technology mapping. During synthesis and technology mapping, networks of logic gates in a user circuit design 101 are converted into circuit elements that can be placed into a programmable IC. The logic gates can be converted into various types of circuit elements such as lookup tables (LUTs), registers, RAM, and digital signal processing (DSP) blocks, depending on the particular programmable IC architecture.

[0023] During placement step 104, a packing tool of the present invention groups the LUTs, registers, and other circuit blocks generated during technology mapping into logic blocks. Each logic block typically includes at least one LUT and at least one register. Logic blocks can be grouped into clusters of logic blocks such as a LAB on a programmable IC. A placement tool then places these logic blocks into actual logic blocks based on the geography of a programmable IC.

[0024] The next step 104 involves routing. During routing, the LABs and logic blocks are connected together by programming the routing resources.

[0025] The placement of the circuit blocks can greatly effect timing and routability of the user made design. Logic blocks and LABs that interconnect should be placed close to each other on the programmable IC to reduce the demand on the routing resources and to reduce signal propagation times. As discussed above, prior art tools grouped LUTs and registers into logic blocks that could not be separated during placement to improve timing and routability of the user design.

[0026] The present invention solves this problem by separating each LUT and register into separate abstract blocks that can be subsequently broken up. Figure 2 graphically illustrates an embodiment of the present invention. A packing tool of the present invention separates LUT1, LUT2, register 1 and register 2 into separate abstract blocks as shown in the upper left corner of Figure 2. The packing tool then groups the LUTs and registers into a logic array block (LAB) 201. The LAB includes several logic blocks.

[0027] Some of the LUTs and registers can be combined into the same logic blocks as shown in Figure 2. For example, LUT 1 and register 2 are both grouped inside logic block LB1. In the Figure 2 example, the packing tool places at least one logic block and at least

4

one register inside each logic block. Even though some of the LUTs and registers are grouped into the same logic block, the present invention treats each LUT and each register as a separate block so that they can be moved into different logic blocks at anytime.

[0028] The present invention allows circuits (e.g., LUTs and registers) that are grouped into one logic block to be separated and regrouped into separate logic blocks or separate LABs before, during, or after placement and routing. A placement tool of the present invention has the ability to separate a LUT and a register in a logic blocks to improve the routability and timing of a user design for a programmable IC.

[0029] Figure 2 illustrates an example of how a placement tool can separate a LUT and a register that have been grouped into the same logic block. As shown in Figure 2, LUT1 and register REG2 were both placed inside logic block LB1 of LAB 201 initially.

[0030] A placement tool may subsequently determine that the demand on the programmable interconnect routing resources would be significantly reduced if REG2 were moved into LAB 202. After making this determination, the placement tool moves REG2 to LB4 in LAB 202. The placement tool can also move LUT1 to a different logic block or a different LAB to further improve routability and/or timing. In the example of Figure 2, the placement tool moves LUT1 to logic block LB2.

[0031] All design rules of the programmable IC should be satisfied before a LUT or a register is grouped into a logic block and a LAB. Figure 3 illustrates a process that can be implemented by a packing tool according to an embodiment of the present invention. In the process of Figure 3, abstract blocks are grouped into clusters of logic blocks (e.g., a LAB) only if the design rules are satisfied. Each abstract block includes only one circuit element, for example, one LUT, one register, one DSP block, one RAM block, or one of another type of circuit block. The abstract blocks are grouped into the clusters based on timing and routing issues.

[0032] The process illustrated in Figure 3 is merely one example of the present invention, and is not intended to limit the scope of the present invention. The process of Figure 3 is now discussed in detail.

[0033] As discussed above, technology mapping generates a netlist 301 of lookup tables, registers (e.g., flip-flops), and other circuits that perform functions in a user made design for a programmable IC. Each circuit is grouped into a separate abstract block. The abstract

5

blocks are not associated with clusters of abstract blocks, and are therefore marked as unclustered. Netlist 301 is the input to the process of Figure 3.

[0034]   At decisional step 302, the packing tool determines if any of the abstract blocks in netlist 301 are not associated with a cluster. Because all of the abstract blocks that are in netlist 301 are initially unclustered, the packing tool proceeds to step 303. At step 303, the tool creates a new cluster. A cluster corresponds to a group of configurable logic blocks on a programmable IC, such as a logic array block (LAB).

[0035]   At decisional step 304, the packing tool determines whether the cluster created at step 303 is full. At this point, the cluster has just been created, so it not full. At step 305, the packing tool locates an abstract block in netlist 301 that has the greatest attraction to the cluster created at step 303. Attraction is a function of the effect that placing the abstract block inside the cluster has on the routing resources (i.e., routability). Attraction is also a function of the timing constraints placed on the user made design that would be caused by placing the abstract block inside the cluster. Equation (1) is an example of how the attraction of an abstract block to a cluster can be calculated.

$$attraction(block) = (\alpha * routability\_attraction(block) + ((1-\alpha) * tim\_attraction(block)) \quad (1)$$

[0036]   The variable "routability_attraction(block)" is a variable that indicates how many nets and connections of nets will be absorbed into a cluster if the abstract block is placed inside the cluster. The routability_attraction variable is greater for abstract blocks that will result in more nets being absorbed into the cluster. To a lesser extent, the variable "routability_attraction" is greater for abstract blocks that result in more connections of nets being absorbed into a cluster.

[0037]   A net (or network) includes the connections from one terminal to one or more other terminals. For example, an output of a LUT may be connected to an input of register, an input of a multiplier, and an input/output pin. The net for the output of the LUT includes all three of the connections to the register, the multiplier, and the I/O pin.

[0038]   The variable "tim_attraction(block)" is a variable that indicates how many timing critical connections are absorbed into a cluster if the abstract block is placed inside the cluster. The tim_attraction(block) is greater for abstract blocks that will result in a timing critical connection being absorbed into the cluster. When a connection is absorbed into a cluster, that connection can use fast local routing resources.

6

[0039]   The α variable is a tradeoff factor between routability and timing. The tradeoff factor can be varied between 1 and 0 depending on the relative importance of routability for specific user designs.

[0040]   After the abstract block that has the greatest attraction to the cluster is located, the packing tool selects a logic block within the cluster to place the abstract block in, at step 306. Figure 2 illustrates how the packing tool places abstract blocks such as LUTs and registers into logic blocks within a LAB cluster. At decisional step 307, the packing tool determines whether placing the abstract block into the logic block selected at step 306 violates any design rules of the programmable IC. At decisional step 308, the packing tool determines whether placing the abstract block into the cluster violates any of the design rules.

[0041]   Because each abstract block in netlist 301 has only one circuit element (e.g., a LUT or register), the determination as to whether the abstract block can be legally placed into the cluster and the logic block is more complex. The added complexity is the result of the fact that logic blocks have their own set of design rules independent of the design rules for clusters. All of the design rules for the logic block and the cluster must be satisfied before the abstract block is added to the logic block and the cluster.

[0042]   Figure 4 shows an example of a portion of a logic block. The logic block portion shown in Figure 4 illustrates examples of design rules that must be satisfied before an abstract block is placed into the logic block. The logic block of Figure 4 includes eight 3-input LUTs and 10 multiplexers that route output signals of the LUTs. The logic block also has two adder circuits 401 and 402 that are coupled together. Adder circuit 401 receives a carry input signal CIN from an adjacent logic block, and adder circuit 401 generates a carry output signal COUT that is transmitted to another adjacent logic block.

[0043]   The logic block of Figure 4 receives at most 8 unique input variables a, b, c1, c2, d1, d2, e and f. This logic block can be configured to implement several different functions. For example, the logic block can be configured as two 5-input or one 6-input lookup table. When implemented as two LUTs, the logic block of Figure 4 implements two functions and generates two output signals.

[0044]   As specific example, the logic block of Figure 4 can be implemented as two 5-input LUTs that implement addition functions and that generate two output signals. Adder circuit 401 generates the first output signal, which is the addition of function $z_0(a, b, c1, d1)$ and

7

function $z_1(a, b, c1, e)$. Adder circuit 402 generates the second output signal, which is the addition of function $z_5(a, b, c2, d2)$ and function $z_6(a, b, c2, f)$.

[0045] The logic block of Figure 4 can also be configured as two 5-input lookup tables that do not use the adder circuits. In this example, the first lookup table generates output signal $z_2(a, b, c1, d1, e)$, and the second lookup table generates output signal $z_4(a, b, c2, d2, f)$. As a third example, the logic block of Figure 4 can be configured as a single 6-input lookup table that receives 6 unique input variables and outputs the result of function $z_3(a, b, c1, d1, e, f)$. Each of the output signals of the logic block is stored in a register (not shown).

[0046] Other LUT configurations for this logic block shown in Figure 4 are possible if they meet the following requirements. First, the logic block can implement two LUTs, if the LUTs together have no more than 8 unique input variables. Second, if either one of the two LUTs is a six input LUT, both of the LUTs must perform the same function (i.e., have the same LUT mask).

[0047] The logic block of Figure 4 is an example of a logic block that can implement the function of one LUT or two LUTs. The flexibility of this logic block adds complexity to the packing process. If the abstract block selected at step 305 is a LUT, a packing tool of the present invention determines whether the LUT can be placed into particular logic blocks based on how many input variables the LUT has.

[0048] For example, if the LUT has 6 inputs, it can only be grouped into a Figure 4 type logic block that does not already have a LUT. Alternatively, a 6-input LUT can be placed into a Figure 4 logic block, if that logic block already includes a LUT that performs the same function as the LUT being considered, and shares at least four of the same input signals. At step 307, the packing tool determines whether a 6-input LUT grouped into a logic block at step 306 satisfies these design rules.

[0049] Two 5-input LUTs can be grouped into the Figure 4 logic block if the LUTs have 2 of the same input variables. The LUT block selected at step 305 cannot be grouped into a logic block if the LUT does not have at least 2 input variables in common with a LUT already grouped into that logic block. The packing tool checks whether this design rule is satisfied at step 307.

[0050] Two 4-input LUTs can be grouped into the Figure 4 LE. At step 307, it is automatically legal to group two 4-input LUTs into the LE.

8

[0051] These are merely a few examples of the types of design rules that can govern the grouping of LUTs into logic blocks. Design rules are generally limited to the particular architecture of the logic block. The techniques of the present invention are applicable to many other types of architectures with different sets of design rules.

[0052] If grouping an abstract block into a logic block does not satisfy one of the design rules, the packing tool then determines if there are any other logic blocks in the cluster that the abstract block can be placed into, at decisional step 310. If there are additional logic blocks remaining, the logic block is selected at step 306, and then the packing tool determines whether grouping the abstract block into that logic block satisfies all of the logic block's design rules at step 307.

[0053] At decisional step 308, the packing tool determines whether it is legal to place the abstract block inside the cluster that includes the logic block selected at step 306. The packing tool makes the determination of whether it is legal to place a group of abstract blocks inside a cluster by counting how many different resources the cluster requires. For example, a cluster may be restricted to two clock inputs. In this example, the packing tool checks to make sure that the group of abstract blocks trying to go into a cluster does not use more than two distinct clocks in order to maintain a legal cluster.

[0054] In general, the legality checks performed at step 308 determine if grouping the abstract block into the cluster creates demands that exceed the cluster's resources. For example, placing an abstract block into a logic block may require more routing resources than the cluster has available and that have not already been assigned to other circuits.

[0055] Legality checks for clusters are typically not related to the specific ordering of logic blocks within the cluster. In general, rearranging the logic blocks in the cluster in any way does not affect the legality of that cluster. The order independence of the logic blocks is guaranteed by the legality checks that are performed at step 307. By ensuring that the abstract blocks are legally placed within logic blocks at step 307, the logic blocks can be rearranged within the cluster without effecting the legality of that cluster.

[0056] If the abstract block can be legally placed into the cluster, the packing tool adds the abstract block to the cluster at step 309, and the position of the logic block in the cluster is recorded (e.g., the logic block number and location of the abstract block within the logic block). Steps 304-310 are then repeated for additional abstract blocks until the cluster is full.

9

[0057] If an abstract block cannot be legally placed into the cluster, the packing tool determines if there are more logic blocks in the cluster that have not been considered at step 310. If there are more logic blocks left to consider, the tool selects one of the remaining logic blocks at step 306 and processes the legality checks of steps 307-308 again.

[0058] Once all of the logic blocks have been considered, the packing tool determines whether there are any abstract blocks left to consider at step 311. At step 311, the packing tool determines if any of the abstract blocks that have not been placed in a cluster have been considered for possible addition to the cluster. If there are abstract blocks remaining that have not been considered for addition to the cluster, the packing tool looks for the abstract block with the greatest attraction to the cluster at step 305, if the cluster is not full.

[0059] If there are no more remaining abstract blocks left to consider, the packing tool returns to step 302 to determine if there are any more unclustered abstract blocks remaining. If unclustered abstract blocks remain, the packing tool creates a new cluster and subsequent steps are repeated as discussed above. If there are no additional unclustered abstract blocks remaining at step 302, the packing tool terminates at step 312 and outputs a list of abstract blocks that have been grouped within each cluster.

[0060] According to further embodiments of the present invention, steps 307 and 308 can be omitted if the programmable IC architecture has only one type of LUT and one type of register in each logic block. In this embodiment, each of the LUT and register blocks in the netlist can be legally placed into any of the logic blocks, because the programmable IC only has one type of logic block.

[0061] The present invention also applies to programmable IC architectures that do not have LABs or clusters of logic blocks. According to these embodiments, the packing tool groups the LUT and register blocks into logic blocks, but not into clusters of logic blocks. The packing tool can also check whether the LUT or register can be legally grouped into a particular logic block. In these embodiments, a packing tool selects a logic block to put an abstract block in based on the attraction of the abstract block to the logic block.

[0062] By using a more abstract block that includes only one circuit element, the present invention optimizes the process of mapping LUTs, registers, and other circuit elements to logic blocks and clusters of logic blocks. During or after placement, a placement tool or routing tool may determine that an abstract block grouped into a cluster would be better placed into another cluster to save routing resources and/or to improve signal timing. The

present invention allows an abstract block containing a single basic circuit element to be placed into a different logic block and/or a different cluster.

[0063] It is highly advantageous to provide this added flexibility, because the placement and routing processes reveal routing and timing concerns that are not apparent previously. The present invention allows the grouping decisions made during the process of Figure 3 to be reversed so that an abstract block can be placed into a different logic block and/or a different cluster. The abstract blocks can be rearranged into different logic blocks and different clusters before, during, or after the placement process. The same legality checks discussed above are implemented to ensure the new positions of the abstract blocks are allowed by the architecture of the programmable IC. If an abstract block can be rearranged in a way that is legal and that would result in an overall beneficial effect on the placement of the user design, a packing tool of the present invention rearranges the abstract block.

[0064] Figure 5 illustrates an example of a process of the present invention for grouping abstract blocks into logic blocks and separating abstract blocks from the logic blocks to improve design goals. At step 501, each circuit element is assigned to a separate abstract block. At step 502, each of the abstract blocks is grouped into a configurable logic block.

[0065] At decisional step 503, a determination is made whether placement information indicates that a design goal would be improved by rearranging at least a portion of the user design. The user design can be rearranged by moving one or more of the abstract blocks into different logic blocks than the ones they were grouped with in step 502.

[0066] Examples of design goals include signal timing and routability. Routability is generally improved by reducing a user design's use of routing resources. Signal timing is improved by reducing signal delay times in a user design.

[0067] Placement information includes geographical information that is gathered by attempting to locate at least a portion of the user design on the programmable integrated circuit (IC). Placement information can include data that is obtained by placing a portion of or all of a user design onto a programmable IC using a placement tool. Placement information can include information obtained from the placement of logic blocks onto a programmable IC.

[0068] Placement information can also include information obtained by floorplanning or partitioning. Floorplanning is performed by assigning blocks of a user design to regions of a

11

programmable IC. Partitioning is performed by assigning half of a user design to one half of a programmable IC, and assigning the other half of the user design to the other half of the programmable IC. Each partitioned half of the user design can be further partitioned and assigned to smaller regions of the programmable IC. Partitioning and floorplanning do not

5   require placing logic blocks of the user design onto the programmable IC.

[0069]   If the placement information indicates that a design goal would be improved by placing at least one of the abstract blocks into a different logic block, that abstract block is removed from its logic block at step 504, and placed into a different logic block at step 505.

[0070]   The techniques of the present invention can be implemented to place user design for

10   many different types of programmable ICs. Figure 6 illustrates an example of one type of programmable IC architecture that can be used with the techniques of the present invention. PLD 600 includes a two-dimensional array of programmable logic array blocks (or LABs) 602 that are interconnected by a network of column and row interconnects of varying length and speed. LABs 602 include multiple (e.g., 10) logic elements (or LEs). An LE is a type of

15   logic block.

[0071]   PLD 600 also includes a distributed memory structure including RAM blocks of varying sizes provided throughout the array. The RAM blocks include, for example, 512 bit blocks 604, 4K blocks 606, and a MegaBlock 608 providing 512K bits of RAM. These memory blocks can also include shift registers and FIFO buffers. PLD 600 further includes

20   digital signal processing (DSP) blocks 610 that can implement, for example, multipliers with add or subtract features. I/O elements (IOEs) 612 located, in this example, around the periphery of the device support numerous single-ended and differential I/O standards. It is to be understood that PLD 600 is described herein for illustrative purposes only and that the present invention can be implemented in many different types of PLDs, FPGAs, and the like.

25   [0072]   While PLDs of the type shown in Figure 6 provide many of the resources required to implement system level solutions, the present invention can also benefit systems wherein a PLD is one of several components. Figure 7 shows a block diagram of an exemplary digital system 700, within which the present invention can be embodied. System 700 can be a programmed digital computer system, digital signal processing system, specialized digital

30   switching network, or other processing system. Moreover, such systems can be designed for a wide variety of applications such as telecommunications systems, automotive systems, control systems, consumer electronics, personal computers, Internet communications and

12

networking, and others. Further, system 700 can be provided on a single board, on multiple boards, or within multiple enclosures.

[0073] System 700 includes a processing unit 702, a memory unit 704 and an I/O unit 706 interconnected together by one or more buses. According to this exemplary embodiment, a programmable logic device (PLD) 708 is embedded in processing unit 702. PLD 708 can serve many different purposes within the system in Figure 7. PLD 708 can, for example, be a logical building block of processing unit 702, supporting its internal and external operations. PLD 708 is programmed to implement the logical functions necessary to carry on its particular role in system operation. PLD 708 can be specially coupled to memory 704 through connection 710 and to I/O unit 706 through connection 712.

[0074] Processing unit 702 can direct data to an appropriate system component for processing or storage, execute a program stored in memory 704 or receive and transmit data via I/O unit 706, or other similar function. Processing unit 702 can be a central processing unit (CPU), microprocessor, floating point coprocessor, graphics coprocessor, hardware controller, microcontroller, programmable logic device programmed for use as a controller, network controller, and the like. Furthermore, in many embodiments, there is often no need for a CPU.

[0075] For example, instead of a CPU, one or more PLDs 708 can control the logical operations of the system. In an embodiment, PLD 708 acts as a reconfigurable processor, which can be reprogrammed as needed to handle a particular computing task. Alternately, programmable logic device 708 can itself include an embedded microprocessor. Memory unit 704 can be a random access memory (RAM), read only memory (ROM), fixed or flexible disk media, PC Card flash disk memory, tape, or any other storage means, or any combination of these storage means.

[0076] While the present invention has been described herein with reference to particular embodiments thereof, a latitude of modification, various changes, and substitutions are intended in the present invention. In some instances, features of the invention can be employed without a corresponding use of other features, without departing from the scope of the invention as set forth. Therefore, many modifications may be made to adapt a particular configuration or method disclosed, without departing from the essential scope and spirit of the present invention. It is intended that the invention not be limited to the particular

13

embodiment disclosed, but that the invention will include all embodiments and equivalents falling within the scope of the claims.